第5章

オンチップ・バスを利用したシステムLSIの最適化設計

――IPコアの処理データや外部I/Fを考慮してバス・アーキテクチャを選ぼう

大庭信之

ここでは，バス・インターフェースの設計を取り上げる．システムLSIでは，IPコアどうしを接続するとき，標準のオンチップ・バスを利用することが少なくない．こうしたバスを用いるには，仕様をきちんと把握し，その特徴を生かすことが重要になる．ここでは，米国IBM社のCoreConnectを例に，バス・アーキテクチャや設計事例を紹介する．（編集部）

近年，半導体技術の進歩とともに，一つのLSIにより多くの機能が集積できるようになってきました．さまざまなハード・マクロとマイクロプロセッサを混載したシステムLSI（SOC：system on a chip）が注目されています．システムLSIの設計において，性能や機能，タイミング仕様などの問題を解析するには多くの時間を要しますが，規格化されたバス（オンチップ・バス）を利用することでこうした時間の短縮を期待できます．また，規格化されたバスを利用すると，設計者は製品を差異化する作業に集中できます．そのため，複雑なシステムLSIの設計にかかるコストや開発期間を削減することが可能です．

注1：本稿はPLBバージョン4.6をベースに解説している．CoreConnectの規格は年々進歩しているので，実際に設計するときは参考文献(1)などから最新情報を入手してほしい．

現在，オンチップ・バスとしていくつかの規格が存在しますが，ここでは米国IBM社の「CoreConnect」注1というバス規格を取り上げながら，バス・インターフェースの設計について解説します．なお，ここで取り上げているのは本オンチップ・バスの特徴的な仕様のみです．

1 基本アーキテクチャを把握する

CoreConnectは1996年にIBM社が発表したLSIチップ内部のバス規格で，IPコアをチップ内でつなぎ合わせるときのベースになるものです．CoreConnectアーキテクチャは，PLB（Processor Local Bus），OPB（On-chip Peripheral Bus），バス・ブリッジ，DCR（Device Control Register）バスから構成されます（**図1**）．おおまかに分類すると，高速・広帯域・低遅延（レイテンシ）を必要とするシステム・コアはPLBに，そうでないものはOPBに接続されます．DCRバスは初期設定や，ふだん設定を変更しない機能を制御するために用いられます．DCRバスはPCIバスのコンフィグレーション・サイクルに近いものです．

PLBは高速ですが，すべてのコアをPLBに接続すると，動作の遅いコアにひっぱられてしまい，動作の速いコアの

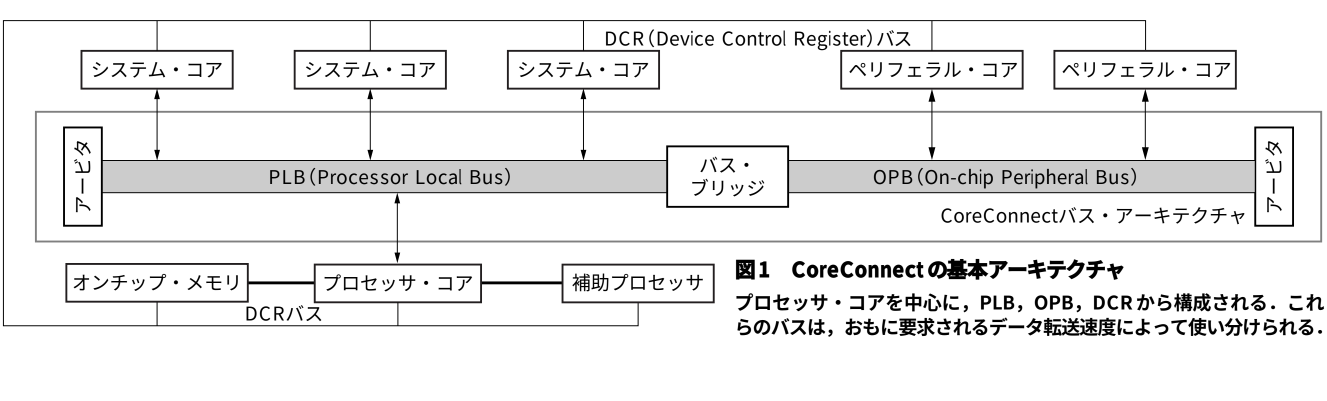


図1　CoreConnectの基本アーキテクチャ
プロセッサ・コアを中心に，PLB，OPB，DCRから構成される．これらのバスは，おもに要求されるデータ転送速度によって使い分けられる．

性能を発揮できなくなります．そこで，動作の遅いコアはOPBに接続し，PLBとは別のバス・ブリッジを介してデータ転送を行います．こうすることで，PLB上のトラフィックが抑えられ，システム全体の性能が向上します．PLBとOPBは同期型バスです．DCRバスもベースは同期型バスですが，ハンドシェイクは非同期的な動きをします[注2]．

● ドライブ信号をOR接続してデータ・バスを制御

ここで，本オンチップ・バスで使われる信号名の標準表記方法について触れておきます(**表1**)．その信号がどの部分に使われているか，だれが出力しているかなどがわかるように，すべての信号にPrefix(接頭語)が付いています．

本オンチップ・バスがPCIなどのボード間やチップ間をつなぐバスと異なる点は，複数コアのコマンド，アドレス，データ信号などをバス上で結線する方法にあります．PCIなどでは，バスをドライブしていないデバイスはI/O出力をハイ・インピーダンスにします．あるいは，オープン・コレクタを用います．一方，本オンチップ・バスでは，複数のドライブ信号をORで接続します．言い換えると，バスをドライブしていないコアは‘0’を出力します．ドライブするコアは，‘0’または‘1’をドライブし，それらをすべてORしてバス信号とします．

図2は二つのOPBマスタ・コアと二つのOPBスレーブ・コアをつないだ例です．それぞれのコアはDBusEn信号でバスをドライブしています．コアからドライブされるデータ・バス(DBus)がすべてORされているので，DbusEn信号が‘0’，すなわちバスをドライブしていないときは，データ・バスの値は‘0’に固定されます．

バスにつながるコアの数が多くなればなるほどORする信号数が多くなり(OR木が広く深くなり)，バスの動作速度が低下することになります．それでも，実際はコアごとにバスをドライブするタイミングが異なるため，速いコアの信号はOR木の枝のほうへ，遅いコアの信号はOR木の根元のほうへつないでバス・タイミングを最適化するという手段をとることもあります．

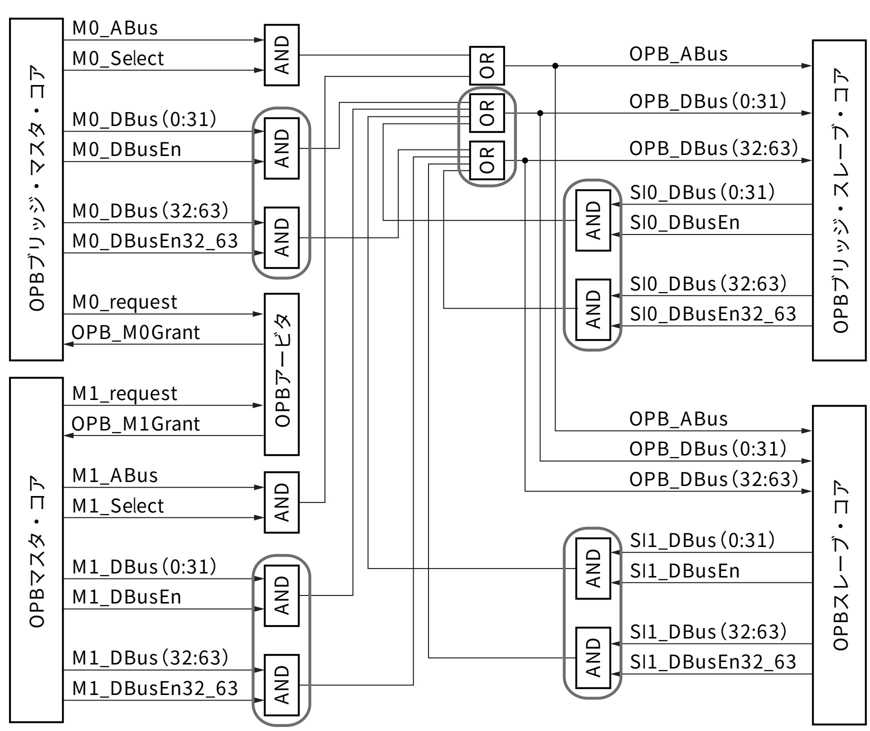
2 高速データ転送向けPLBの基本サイクル

PLBは性能を高めるために以下の特徴を備えています．まず，リード(読み出し)用とライト(書き込み)用に別々のデータ・バスが用意されており，リード・データとライト・データを同時にバス上に流すことができます．また，アドレス・バスとデータ・バスは物理的に独立しており，アドレス・パイプラインに対応しています．例えば，バス・マスタAがバス・スレーブXへリード・アクセスのためのアドレスを流します．そのデータが返ってくる前に，

注2：本オンチップ・バス規格では，PLB，OPB，DCRそれぞれのバスに合わせて，カスタム回路を設計するためのバス・モデル・ツールが用意されている．バス・モデルは，設計するコアに対して，指定のアクセスを発生させて，さまざまなハンドシェイクを行う過程においてプロトコル誤りがないかどうかをチェックする．詳しくは参考文献(1)を参照のこと．

表1　CoreConnect信号の表記法

Prefix	信号の意味
Mn_	バス・マスタがドライブしている信号 M0_xxxxのようにMの後に数字を置いてバス・マスタIDを示すことがある
Slm_	バス・スレーブがドライブしている信号 Sl0_xxxxのようにSlの後に数字を置いてスレーブIDを示すことがある
PLB_	PLB上のバス信号
OPB_	OPB上のバス信号



▶ 図2
バス上の信号結線
DBusEnはデータ・バスをドライブするときに‘1’になる．ドライブしないときは‘0’になり，ORへの入力をすべて‘0’にする．

バス・マスタBがバス・スレーブYへリード・アクセスのためのアドレスを流すことができます．

バスのアービトレーションは，データ転送が行われている最中にオーバラップして行うことができます．

また，柔軟にシステムを構成できるように，以下の機能を備えています．

- 16個までのバス・マスタに対応．バス・スレーブ数は制限なし
- バス・マスタごとに4段階のバス優先度を設定可能．バス・リクエスト時に動的に変更可能
- 32/64/128/256ビットのデータ・バスを選択可能
- バス・ロック機能で不可分アクセスを実現
- バイト単位の転送や，複数ワードを一度に転送するバースト・アクセスに対応

図3はマスタ，スレーブ，アービタの接続概念図です．PCIバスやISAバスなどとは違い，マスタとPLBコアの間の信号の多くが1対1接続になっていることがわかります．

● 読み出しと書き込みのサイクルは微妙に異なる

以下に，タイミング・チャートを使ってPLB上のさまざまなサイクルを説明します．

図4に基本的なリード・サイクルを示します．信号名とその役割については，**表2**を参照してください．**図4**におけるサイクルごとの動作は，以下のようになります．

- **サイクル1** ──バス・マスタはリードを行うアドレスやバイト，サイズなどを指定して，バス・リクエスト（Mn_request）を出す．同時に，Mn_priorityによってバス要求優先度（低，中，高，最高）をサイクルごとに指定できる（バス使用の緊急度に合わせたバス・システムを実現できる）．この例では，バスが空いているので，バス・アービタはすぐにPLB上にアドレスを流し，PLB_PAValid信号をアサートする．また，ターゲット・スレーブはアドレスをデコードし，サイクル1ですぐにSl_waitを返している．
- **サイクル4** ──スレーブがアドレスを受け取ったので，Sl_AddrAckを返してアドレス・サイクルが終了する．
- **サイクル5** ── PLB_PAValidにおいて“Next Wr Avalid”は，「次のライト・アクセスのためのアドレス・サイクルは，最速でこのサイクルから開始できる」という意味．PLBはリード用とライト用に別々のデータ・バスを持っているので，このようなことが可能となる．

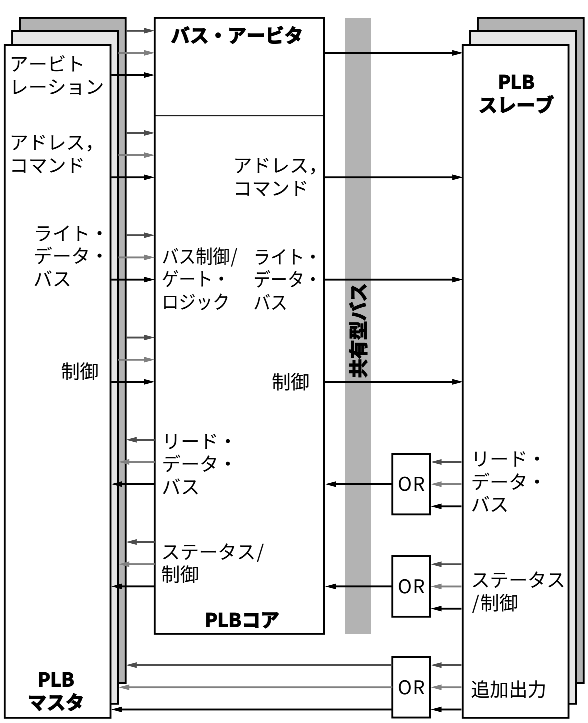


図3　PLBバス・マスタ，スレーブ，バスのハードウェア構成

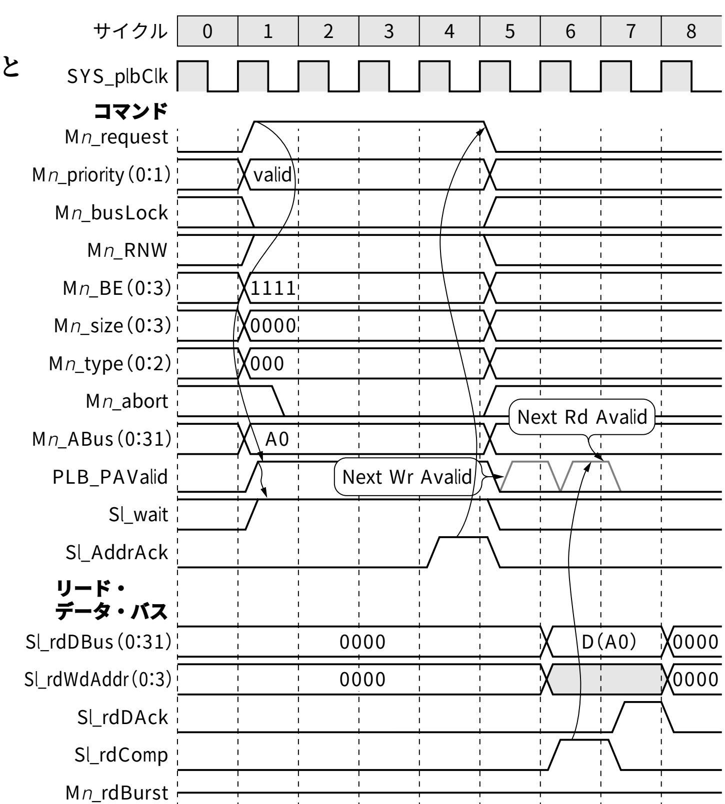


図4　基本リード・サイクル（アドレス・パイプラインなし）

アドレス・サイクルはSl_AddrAckで，データ・サイクルはSl_rdCompで終了する．実際のデータはSl_rdDAckで流れることに注意．

- **サイクル6** ──スレーブがリード・データをリード・データ・バス上に乗せ，Sl_rdCompをアサートする．Sl_rdCompは最後のリード・データ・サイクルの1サイクル前にアサートされる．これは，次のリード・サイクルが空きサイクルなしに実行できるようにするためのくふう．“Next Rd Avalid”は，「次のリード・サイクルは，最速でこのサイクルから実行できる」という意味．
- **サイクル7** ──スレーブがSl_rdDAckをアサートし，バスを介してデータがマスタへ転送される（この例ではサイクル6においてすでにデータがバスに乗っているが，サイクル7だけでも問題ない）．これでリードが終了する．

図5は基本ライト・サイクルのタイミング・チャートです．リード・サイクルと違う点は，Mn_RNWが“L”になっていることと，Mn_requestと同じタイミングでマスタからのライト・データがMn_wrDBus上に出始めるということです．このライト・データは，Sl_wrDAckが返ってくるまでマスタから出し続けなければなりません．

表2　PLBの信号とその役割

信号名	役　割
Mn_request	バスの使用権を要求する
Mn_priority	Mn_requestと同時に出力し，バス要求の優先度を指定する．“11”はもっとも優先度が高いことを，“00”はもっとも優先度が低いことを表す
Mn_busLook	不可分サイクルを実行したいときなど，バスを使用し続けたい場合にアサート
Mn_RNW	リードかライトかを指定する．“H”ならリード，“L”ならライト
Mn_BE(0：3)	バイトを指定する．本文の例では，32ビット・データ・バスを使っているので，4本になる
Mn_size(0：3)	バースト転送時のサイズを指定する．主な指定として0000：1回，0001：4回，0010：8回，0011：16回がある
Mn_type(0：2)	転送タイプを指定する[1]
Mn_about	バス・マスタが途中でサイクルを終了したいときに使用する
Mn_Abus(0：31)	アドレス・バス
PLB_PAValid	アドレス・バス上に有効なアドレスが乗っていることを示す
Sl_wait	スレーブがアドレス・サイクルを引き伸ばしたいときに使用する．PLBには，バス・マスタが誤ってスレーブが割り当てられていないアドレスにアクセスした場合，バスがデッド・ロックすることを防ぐためのタイムアウトのしくみがある．Sl_AddrAck信号を出さなければアドレス・サイクルは延長されるが，このSl_waitをアサートしておかないとアドレス・サイクルが16サイクルで，自動的にタイムアウトしてしまう
Sl_addrAck	スレーブがアドレスをゲットしたことを示す
Mn_wrDBus(0：31)	ライト用データ・バス
Sl_wrDAck	ライト・データのアクノリッジ(Ack)
Sl_wrComp	ライト・データ・サイクルの終了を示す
Mn_wrBurst	ライト・サイクルがバーストであることを示す
Sl_rdDBus(0：31)	リード用のデータ・バス
Sl_rdWdAddr(0：3)	バースト転送時のワード・アドレスを示す
Sl_rdDAck	リード・データのAck
Sl_rdComp	リード・データ・サイクルの終了を示す
Mn_rdBurst	リード・サイクルがバーストであることを示す

● 読み出しと書き込みのバースト転送を同時に行う

図6はバースト・リードとバースト・ライトが連続して行われた場合のタイミング・チャートです．サイクル1でリードのアドレス・サイクル，続いてサイクル2でライトのアドレス・サイクルがそれぞれスレーブからSl_addrAckを受けて完了します．マスタはMn_sizeを“0001”としてバースト長4を要求しています．この例では，後から始まったライトのデータがサイクル2～5で4回バースト転送されています．その後，リードのデータは1サイクル遅れてサイクル3～6で転送されています．先に述べたようにPLBではリード用とライト用で別のバス線を持っているので，このようにリードとライトのデータを同時に転送することができるのです．

バースト転送については誌面のつごうで詳しくは述べませんが，PLBではこのようにあらかじめバースト長を指定して転送を行うモードのほかに，転送中に動的にバースト

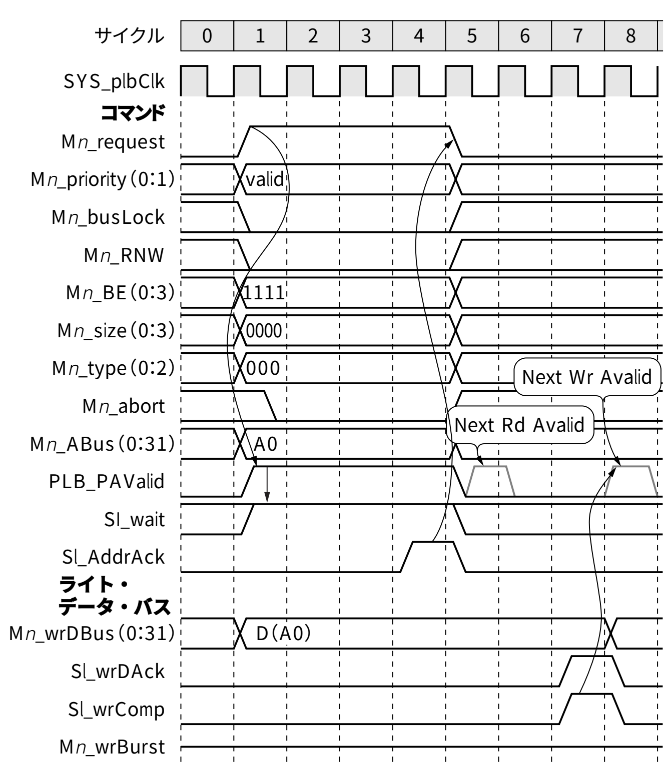


図5　基本ライト・サイクル（アドレス・パイプラインなし）
アドレス・サイクルはSl_AddrAckで，データ・サイクルはSl_wrCompで終了する．ライトの場合，最短でもSl_wrCompとSl_wrDAckが同時に出る．

長を決定するモードがあります．マスタから，あるいはスレーブから転送中止を要求することで，柔軟性の高いバスを構築できます．

● バス占有を防ぐ遅延タイマ

一つのバス・マスタがバスを占有することを防ぐため，4回を超える長いバースト・サイクルを発生するバス・マスタは，遅延（レイテンシ）タイマを装備しなければなりません．

このタイマは，最初のデータ・アクノリッジ（Sl_rdDAck, Sl_wrDAck）が返ってきた時点からカウントを開始します．あらかじめセットされた最大値を超えると，自分以外にバスを使いたがっているマスタがあり，その優先度が自分と同じか高い場合，現在行っているバーストを中断します[注3]．バス上には，PLB_rdPendReq，PLB_rdPendPri，PLB_wrPendReq，PLB_wrPendPriという信号が用意されており，バス・マスタは自分以外にだれかがバスを要求しているかどうか，およびその優先度がいくらかがわかるようになっています．

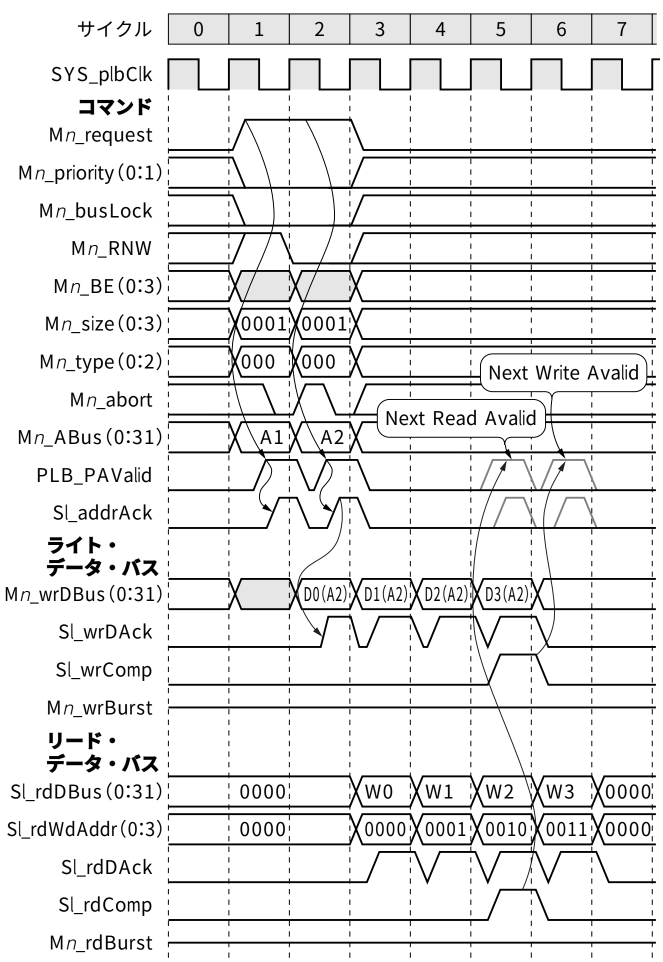


図6　バースト・リード・サイクルとバースト・ライト・サイクル
リード（読み出し）用とライト（書き込み）用で別のバス（ワイヤ）が用意されているので，リード・データとライト・データを同時に転送できる．

● リードとライトを続けて実行するロック付きサイクル

スレッド間の同期などを行うときなど，PLB上でリードとライトを不可分に実行する必要があるときは，バス・ロックを使います．

図7はその一例です．サイクル1でバス・マスタ1が，サイクル2でバス・マスタ2がそれぞれバス要求を出しています．バス・マスタ1はリードとライトを不可分に行うため，M1_busLock信号をアサートしています．このとき，リード・データ・バスもライト・データ・バスも使われていないので，アービタはバス・マスタ1のリード・サイクルをサイクル1から開始します．スレーブはSl_addrAckを同じサイクルで返しています．

バス・ロック信号がなければ，サイクル2でリード・データ・サイクルの終了を知らせるSl_rdCompが出ているので，バス・マスタ2のリード・サイクルを開始します．しかし，**図7**の場合はM1_busLockが出ている（つまりロックされている）ので，そのままバス・マスタ1からのリクエストを待ちます．サイクル4でバス・マスタ1のライト・サイクルのアドレス・サイクルを終了します．サイクル5でM1_busLockが“L”になっているのを確認し，サイクル6でバス・マスタ2のリード・サイクルを開始します．

● アドレス・パイプラインでバスの高速化を図る

バスの性能を向上させる一つの方法としてアドレス・パイプラインがあります．簡単に言うと，これは1番目のリードのデータが返ってくる前に，2番目，あるいは3番目のリード・リクエストをバスに乗せるというものです．

図8を見てください．マスタ0，1，2，3が順番にリード・リクエストをそれぞれ別のスレーブ1，3，5，7に送っています．注目する信号はPLB_SAValidとPLB_rdPrim()です．PLB_SAValidは補助アドレスを意味し，バス上に未終了のサイクル（**図8**の例ではマスタ0が発行したリード・サイクルのデータが返ってきていない）がある場合，PLB_

注3：自分以外のバス・マスタからリクエストがある場合は優先度に関係なくバーストを中断する，リクエストの有無にかかわらずバーストを中断する，というオプションもある．

PAValidの代わりにアサートされます．どちらをアサートするかは，**図3**に示したアービタとPLBコア・ロジックがバス上のサイクルの進みぐあいをモニタし，判断します．データ・バス上を流れるデータの順番はリクエストの順番と同じで，入れ替わることはありません．

スレーブがPLB_PAValidを受けたときはSl_addrAckを返し，リード・データを用意できた時点でリード・データ・バスにデータを乗せてSl_rdComp，Sl_rdDAckを返してもかまいません．しかし，PLB_SAValidで受けたリクエストについては，先に送らなければならない（PLB_PAValidを受けた）データがあるため，すぐにデータを返せません．ここで，いつデータを返してよいかを知る手段が必要となるわけですが，このためにPLB_rdPrimという信号が用意されています．スレーブはアービタからこの信号を受けて，最速2サイクル後にマスタに返すデータをリード・データ・バスに乗せることができます．アドレス・パイプライン動作時にサイクルをトレースするのは，アービタとPLBコアの責任です．PLBでは，パイプラインの深さに制限はありません．

3 比較的低速なOPBの基本サイクル

PLBと比べてOPBはバス信号の本数が少なく，プロトコルも簡単です．**図9**にOPBの基本サイクルを，**表3**に各信号の概要を示します．**図9**の各サイクルの動作は以下のとおりです．

- **サイクル2** ──バス・マスタ1がバスの使用要求を出し，アービタはすぐにバスの使用許可を返す．
- **サイクル3** ──バス・マスタはM1_selectをアサートしてサイクルを開始する．同時にM1_RNWを“H”にすることでリードであることを示す．この例では，M1_fwXferとM1_hwXferが両方とも“H”なので，**表3**から32ビット転送要求であることがわかる．有効アドレスをM1_ABusに乗せる．
- **サイクル4** ──スレーブはリード・データを用意し，データ・バスに乗せる．同時に，Sl2_xferAckをアサート

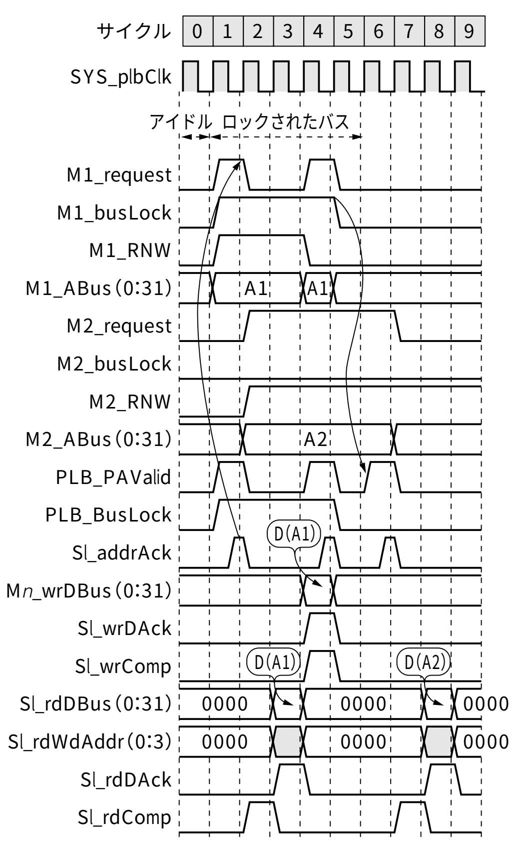


図7　ロック付きサイクル
バスをロック（専有）することで不可分のリード/ライトを行う．

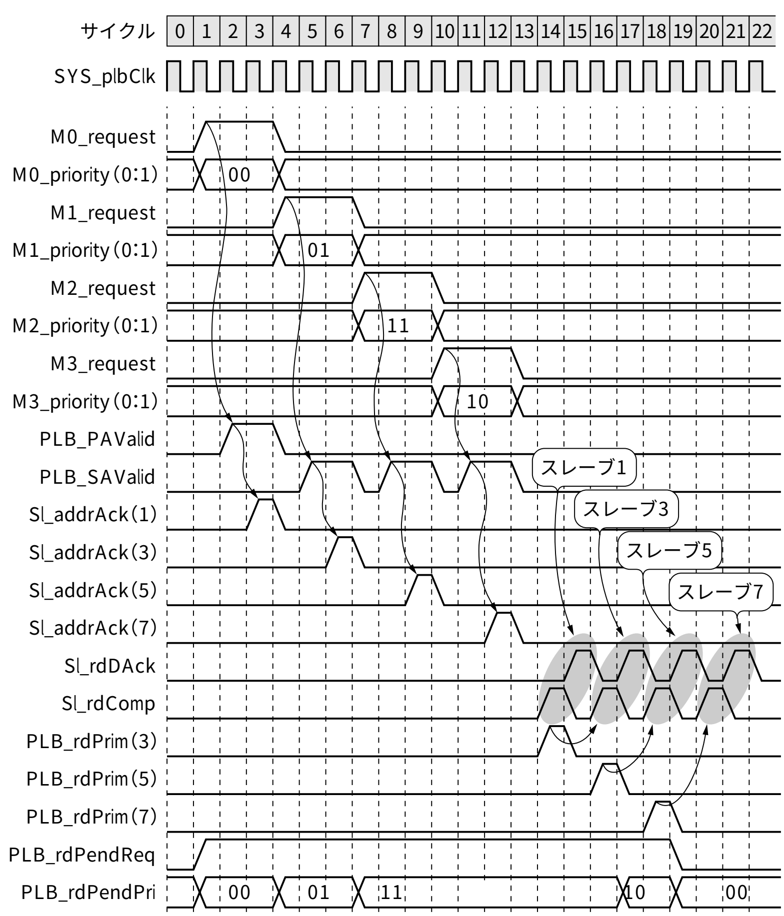


図8　アドレス・パイプライン
リードやライトの遅延（レイテンシ）が長い場合に有効．

することでデータが用意できたことをマスタへ知らせる．

- **サイクル5** ――マスタはSl2_xferAckが返ってきたので，M1_selectをデアサートし，サイクルを終了する．

● 8ビット/16ビットのバス幅のコアと接続する

OPBには比較的低速のコアが接続されます．低速なコアにはデータ・バスが8ビットのものもあるでしょう．OPBでは，ダイナミック・バス・サイジングという機能を使って，32ビット・データ・バスのOPBに8ビット・データを持つコアを接続できるようになっています．

図10はダイナミック・バス・サイジングの一例です．サイクル1において，マスタはM1_fwXferとM1_hwXferの両方を“H”にすることで32ビット・データのリードを開始しています．しかし，スレーブはSl2_fwAck，Sl2_hwAckの両方を“L”にしたままSl2_XferAckを返しています．これは，32ビット・アクセスに対して，1バイト(8ビット)だけB0というデータを返したことを意味しています(**表3**)．

マスタは残りの3バイトを得るため，まずサイクル3で，1バイト(B1)をリードしにいきます．バイト・リードなので，M1_fwXferとM1hwXferは“L”とします．サイクル4では，残り2バイトをハーフ・ワード・アクセスでリードしにいきます．M1_hwXferだけが“H”に変わっていることに注意してください．スレーブは一貫してバイト単位でAckを返しているので，結局32ビット・アクセスは4回の

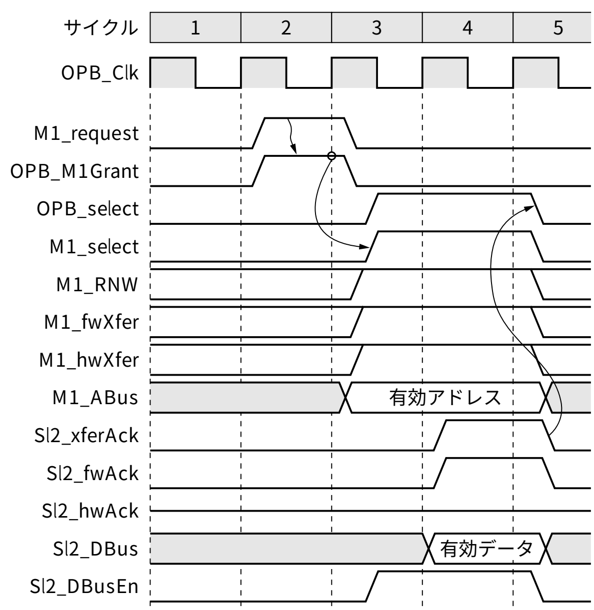


図9　OPB基本サイクル
基本的にはM1_selectとSl2_xferAckのハンドシェイク．

表3　OPBの信号とその役割

信号名	役　割
M1_request	バス・マスタ1のバス使用権要求信号
OPB_M1Grant	OPBアービタからM1へ送るバス使用許可信号
OPB_select	OPB上の転送ストローブ
M1_select	バス・マスタ1が出力する転送ストローブ
M1_RNW	リード/ライトを指定する信号．“H”でリードを，“L”でライトを示す
M1_fwXfer M1_hwXfer	転送サイズを示す M1_fwXfer / M1_hwXfer / サイズ 0 / 0 / 8ビット 0 / 1 / 16ビット 1 / 1 / 32ビット
M1_Abus	アドレス・バス
Sl2_xferAck	転送アクノリッジ(Acknowledge)
Sl2_fwAck Sl2_hwAck	転送アクノリッジのサイズを示す Sl2_fwAck / Sl2_hwAck / サイズ 0 / 0 / 8ビット 0 / 1 / 16ビット 1 / 1 / 32ビット
Sl2_Dbus	データ・バス
Sl2_DBusEn	スレーブがデータ・バスへデータを流すときのゲート

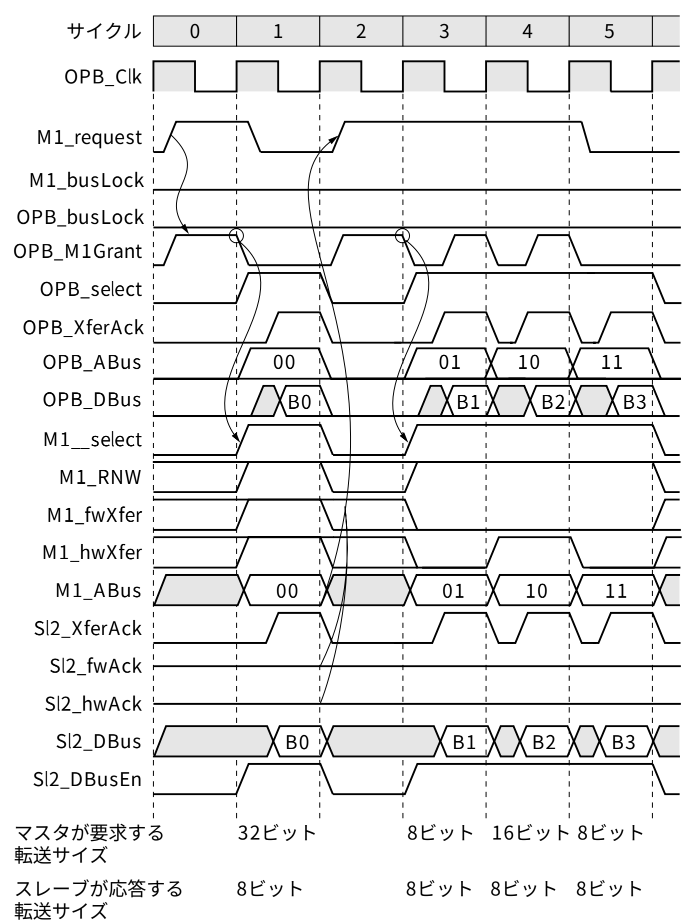
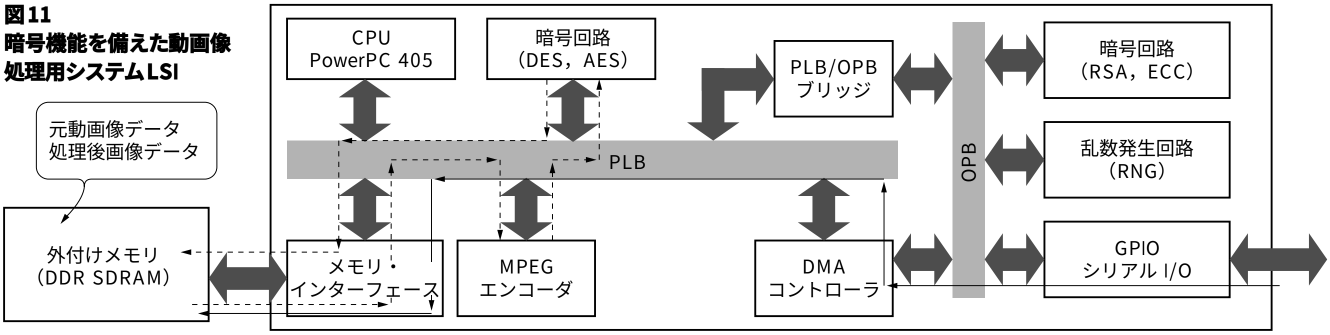


図10　OPBダイナミック・バス・サイジング
最初，マスタは32ビット・アクセスを出してきたが，スレーブが8ビット転送Ackを返しているため，結局4回のバイト転送になっている．

図11
暗号機能を備えた動画像処理用システムLSI



バイト・アクセスに分割されました．このように，マスタはスレーブからのAckを見つつ必要に応じて転送データ・サイズを調整しながらアクセスを繰り返します．

4 暗号付き動画像処理用LSIに応用

最後に，本オンチップ・バスを用いたシステムLSIの設計例として，暗号機能付き動画像処理システムLSIを示します．主な用途は，動画像データをMPEGエンコードして，さらに暗号化するといったことです．

図11に内部構成を示します．CPU（PowerPC）を頭としてPLBに接続され，さまざまなIPコアが処理するデータ量を考慮してPLBとOPBに分けて接続されています．RSA（Rivest Shamir Adleman）/ECC（elliptic curvecryptography）エンジンは公開かぎ暗号用の，DES（data encryption standard）/AES（advanced encryption standard）エンジンは共通かぎ暗号用の演算処理コアです．PLBとOPBはブリッジで相互接続されます．

例として，AさんからBさんへ動画像データを暗号化して送る場合を考えます．おおまかな処理の流れは以下のようになります．

1）Bさんが公開かぎ暗号で用いる公開かぎXをAさんへ送る．公開かぎ暗号の秘密かぎYは自分で持っている．このとき，かぎの生成に乱数発生器を使う．
2）Aさんは動画像データを暗号化するための共通かぎ暗号用共通かぎZを，Bさんから送られてきた公開かぎXを用いて暗号化し，Bさんへ送る．
3）Bさんは公開かぎ暗号用秘密かぎYでZを解く．これで，2人とも暗号化/復号化に用いるZを保持できたことになる．
4）Aさんは動画像データをZで暗号化してBさんへ送る．

公開かぎ暗号のかぎデータは小さい（RSA暗号で1,024～2,048ビット程度）のですが，処理時間がかかります（通常，100万サイクル以上）．加えて，共通かぎを暗号化するので，セッション開始時に1度使うだけです．一方，共通かぎ暗号は大量のバルク・データ（ここでは動画像データ）を高速に暗号化/復号化するときに用います．以上の処理の性質を考慮して，RSA/ECCエンジンと乱数発生器はOPBに，DES/AESエンジンはPLBに接続しています．

簡単に動画像の暗号化処理の流れを説明しておきます．外付けメモリ上にある元動画像データをMPEGエンコーダでMPEGフォーマットへ変換してから，DES/AESエンジンで暗号化し，メモリへ書き戻します．この流れを**図11**の中の点線で示しています．これがこのシステムLSIの主なしごとなのですが，同時に低速の汎用I/OやシリアルI/Oからデータをメモリへ読み書きするというしごともあります．これが**図11**の実線で示した流れです．DMAコントローラは汎用I/OやシリアルI/Oを通して外部デバイスからデータを読み込みますが，外部デバイスが低速なので，データを読み込んでいる最中はOPBを占有します．それでも，PLBは独立して動作できるので，動画像データ処理への影響は最小限に抑えることができます．

このように，システムLSIの内部構成，とくにPLBとOPBの使い分けとIPコアの接続は，処理するデータ量やアクセス頻度，外部インターフェースの性質などを総合的に考慮して決定します．

参考・引用*文献

(1)* IBM Microelectronicsホームページ，http://www.ibm.com/chips/

おおば・のぶゆき
日本アイ・ビー・エム（株）東京基礎研究所

<著者プロフィール>
大庭信之．日本IBM 東京基礎研究所に勤務．計算機アーキテクチャに関する研究に従事している．ooba@jp.ibm.com

5